5К

УДК 004.832

В.В. Краснопрошин, А.В. Карканица
Белорусский государственный университет, г. Минск, Беларусь
Гродненский государственный университет имени Янки Купалы, г. Гродно, Беларусь
krasnoproshin@bsu.by, karkanica@gmail.com

Технология построения динамических предметных областей на основе графовых моделей

В статье рассматривается проблема моделирования динамической предметной области для класса задач, характер и сцена решения которых имеют распределенную структуру. Предлагается способ представления графовой модели предметной области на основе расширения языка разметки направленных графов. Рассматривается вопрос проектирования архитектуры программной системы для решения сложно структурированных задач. В рамках этой архитектуры предлагается реализация программного компонента построения графовой модели динамической предметной области.

Введение

В настоящее время существует большое количество задач, характер и сцена решения которых имеют распределенную структуру. Это задачи выполнения крупных проектов коллективами территориально удаленных исполнителей, задачи управления распределенными организационными структурами, объединенными в некоторую административную иерархию, задачи по разработке программного обеспечения распределенными командами разработчиков и другие [1]. Характерной особенностью этих задач является необходимость распределения работ (декомпозиция задачи) между исполнителями с последующим объединением полученных результатов.

Предметная область описанного класса задач является динамической и распределенной, и возникает актуальная проблема автоматизации процесса формирования предметной области. Решение этой проблемы требует создания программной системы, которая будет формировать единое информационное пространство для команды распределенных исполнителей и обеспечивать их согласованную работу в условиях динамически изменяющейся среды. Для этого на первом этапе необходимо решить задачу построения модели динамической предметной области.

Целью данной работы является разработка технологии построения динамических предметных областей, в частности создание формальной спецификации для описания модели и программная реализация компонента для построения, хранения, визуализации и модификации динамической графовой модели предметной области.

Постановка задачи

Описанный во введении класс задач определим как сложно структурированные задачи (ССЗ). В соответствии с онтологическим подходом модель ССЗ имеет вид [1]:

$$Task = (S, Group, S^1, \ldots, S^n, Req^1, \ldots, Req^k, Solution^1, \ldots, Solution^n, T^1, \ldots, T^n), \quad (1)$$

где S – постановка общей задачи; Group = (Center, $Expert^1$, $Expert^2$, …, $Expert^n$) – группа исполнителей, включающая центр (Center), инициирующий задачу, и распределенных

исполнителей (Expert), реализущих решение подзадач; $S^1, \ldots, S^n$ – постановки подзадач, полученных в результате декомпозиции S; $Req^1, \ldots, Req^k$ – требования к решению; $Solution^1, \ldots, Solution^n$ – формализованная информация, полученная в процессе решения подзадач и в сумме составляющая решение Task; $T^1, \ldots, T^n$ – ограничения временных затрат на решение подзадач; $n \to \infty$, k – переменная величина.

В [2] показано, что соответствующая модель предметной области может быть представлена древовидным ациклическим графом, вершиной которого является задача **S**, узлы определяют иерархию подзадач, дуги – уровень их вложенности.

Пусть G(V,E) – древовидный иерархический граф, представленный множеством вершин E и множеством ребер V. Граф G будем называть динамическим, если за промежуток времени t возможен переход графа из состояния $G^1(V^1,E^1)$ в состояние $G^2(V^2,E^2)$, причем множества (V^1, V^2) и (E^1, E^2) соответственно, не совпадают. Модификацией графа G назовем процесс перехода графа из состояния G^1 в момент времени t_1 в состояние G^2 на момент времени t_2, который может быть вызван выполнением некоторой последовательности операций на графе (добавление вершины, удаление вершины, разбиение графа, слияние графа). В соответствии с (1) каждой вершине графа поставим в соответствие следующий набор атрибутов:

$$v = \langle id, task, name, addr, status, inf \rangle, \qquad (2)$$

где id – уникальный идентификатор вершины (задачи);

task – постановка задачи (требования к решению);

status – состояние вершины (0 – задача инициирована, но не решена; 1 – задача в процессе решения, 2 – задача решена);

name – уникальный идентификатор эксперта;

addr – адрес эксперта;

inf – информационная составляющая (фактически решение задачи, представленное в одном из допустимых форматов).

Требуется:

1. Разработать способ представления графовой модели динамической предметной области задачи (1), позволяющий описать модель в виде формальной спецификации, допускающей машинное представление графа, его автоматизированную обработку и облегченный механизм синтаксического анализа.
2. Реализовать программный компонент для построения, хранения, визуализации и модификации динамических графовых моделей.

Решение первой задачи усложняется тем, что вместе со структурой графа необходимо хранить описание отдельной задачи и набор атрибутов, связанных с каждой вершиной. Для представления таких графов в теории систем искусственного интеллекта разработан язык семантических сетей. Однако для описания иерархически декомпозированных задач такой язык является избыточным, кроме того процедуры вывода по иерархически декомпозированному графу значительно проще процедур вывода по семантической сети. В связи с этим предлагается рассмотреть существующие стандарты описания графовых моделей и возможности расширения их собственного синтаксиса.

Языки описания графовых моделей

Для описания графов и теоретико-графовых моделей в настоящее время существует несколько общепризнанных стандартов.

DOT language – язык описания графов в виде простого текста. Набор атрибутов, применимых к объектам графа, предельно лаконичен, реализация программного интерфейса для работы с ним не представляет особого труда. Однако DOT не предоставляет возможности описывать и добавлять собственные параметры к ребрам и вершинам графа.

GDL (Graph Description Language) – мощный, простой в обращении язык описания графов. Создавать графы в формате GDL можно с помощью любого текстового редактора, поддерживающего сохранение файлов в формате обычного текста.

GraphML (Graph Modeling Language) – полнофункциональный файловый формат для описания графов. Он включает базовый язык, предназначенный для описания структурных свойств графа, поддержку направленных, ненаправленных и смешанных графов, гиперграфов, иерархических графов, описание графического представления, ссылок к внешним данным. GraphML использует синтаксис, основанный на языке XML, и следовательно идеально подходит в качестве универсального средства для представления, хранения и обработки графов.

Однако все представленные форматы не предоставляют гибкого механизма расширения языка собственными свойствами и механизма добавления данных к структурным элементам графа, что не позволяет использовать их для решения задачи представления модели динамической предметной области в виде атрибутированного иерархического графа.

Этого недостатка лишен формат DGML (Directed Graph Murkup Language) – язык разметки направленных графов, который использует простой XML для описания направленного графа. Направленный граф представляет собой набор узлов, соединенных ссылками или границами. Узлы и ссылки могут быть использованы для представления сетевых структур, элементов программного проекта. Можно использовать DGML для визуализации информации, выполнения анализа сложности или просто для просмотра и редактирования графа. В контексте решаемой задачи, существенным является тот факт, что DGML может быть расширен за счет включения структурированных элементов, отвечающих специфике предметной области и за счет добавления дополнительных атрибутов ко всем элементам DGML.

Расширение DGML для описания динамической предметной области

Предлагается использовать базовые элементы DGML, расширив их за счет добавления необходимых дополнительных атрибутов. Основные элементы языка DGML следующие:

– **<DirectedGraph></DirectedGraph>** – корневой элемент графа, все остальные DGML-элементы отображаются внутри области этого элемента;

– **<Nodes></Nodes>** – элемент содержит список элементов, задающих узлы графа;

– <**Links></Links>** – элемент содержит список элементов, задающих ссылки между узлами (ребра графа);

–**<Properties></Properties>** – элемент содержит список элементов, которые используются для присвоения значения любому элементу или атрибуту DGML.

Расширим представленный формат за счет определения дополнительных элементов, которые позволят описать графовую модель в соответствии с моделью ССЗ (1) и учитывая набор атрибутов (2).

Определим элемент **<Node/>** как единичный узел графа, соответствующий подзадаче, элемент **<Link/>** – как единичную ссылку, соединяющую исходный узел с целевым узлом и задающую ребро графа. Тогда элемент **<Node/>** должен содержать набор обязательных атрибутов, позволяющих описать подзадачу, а элемент **<Link/>** – набор обязательных атрибутов, определяющих иерархию подзадач. Реализуем это за счет возможности добавления дополнительных атрибутов к элементам DGML. Для этого определим список элементов <Property/>:

```
<Properties>
  <Property Id=”TaskID” Label=”Идентификатор задачи” DataType=”string” />
  <Property Id=”Status” Label=”Статус задачи” DataType=”int” />
   <Property Id=”NameID” Label=”Идентификатор эксперта” DataType=”string” />
  <Property Id=”Addr” Label=”Адрес эксперта” DataType=”string” />
</Properties>
```

В результате множество вершин графа (задач) описывается в следующем формате:

```
<Nodes>
      <Node TaskID=”…” Status=”…” NameID=”…” Addr=”…”/>
      <Node TaskID=”…” Status=”…” NameID=”…” Addr=”…”/>
      <Node TaskID=”…” Status=”…” NameID=”…” Addr=”…”/>
      …
 </Nodes>
```

Для определения иерархии подзадач элемент <Link/> должен содержать по крайней мере два обязательных атрибута: Source – уникальный идентификатор начальной вершины ребра, Target – уникальный идентификатор конечной вершины. Тогда множество ребер графа может быть представлено следующим образом:

```
<Links>
      <Link Source=”…” Target=”…”/>
      <Link Source=”…” Target=”…”/>
      <Link Source=”…” Target=”…”/>
      ...
</Links>
```

За счет введения дополнительных элементов и соответствующих атрибутов, графовая модель представляется DGML-документом следующего формата:

```
<?xml version="1.0" encoding="utf-8"?>
<DirectedGraph Title="S" >
  <Nodes>
      <Node TaskID=”S1” Status=1 NameID=”E1” Addr=”…”/>
      <Node TaskID=”S2” Status=1 NameID=”E2” Addr=”…”/>
      <Node TaskID=”S11” Status=0 NameID=”E11” Addr=”…”/>
      <Node TaskID=”S12” Status=0 NameID=”E12” Addr=”…”/>
  </Nodes>
  <Links>
    <Link Source=”S” Target=”S1” />
    <Link Source=”S” Target=”S2” />
    <Link Source="S1" Target="S11" />
    <Link Source="S1" Target="S12” />
  </Links>
<Properties>
  <Property Id=”TaskID” Label=”Идентификатор задачи” DataType=”string” />
  <Property Id=”Status” Label=”Статус задачи” DataType=”int” />
  <Property Id=”NameID” Label=”Идентификатор эксперта” DataType=”string”/>
  <Property Id=”Addr” Label=”Адрес эксперта” DataType=”string” />
</Properties>
</DirectedGraph>
```

В результате получена формальная спецификация для представления графовой модели динамической предметной области задачи (1), реализованная за счет использования механизма расширении языка разметки направленных графов DGML.

Архитектура системы

Для практического применения вышеописанных модели предметной области ССЗ и формальной спецификации для представления графовой модели необходимо разработать архитектуру программной системы, специфика которой заключается в распределенности источников информации, необходимых для формирования предметной области. Основное назначение системы – это формирование единого информационного пространства для команды распределенных исполнителей и обеспечение их согласованной работы. Предлагается включить в архитектуру системы компоненты, представленные на рис. 1.

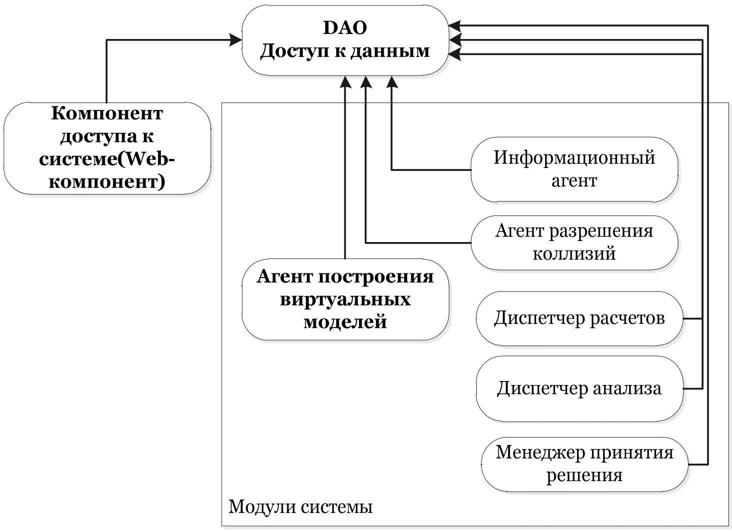


Рисунок 1 – Архитектура программной системы

При разработке архитектуры за основу взят процессный подход [4], который позволяет соотнести процессы решения ССЗ (построение и модификация графовой модели ПрО, добыча экспертных знаний, контроль полноты ПрО и синтез окончательного решения) с соответствующими компонентами программной системы. В результате архитектура имеет многомодульную (компонентную) структуру, каждый компонент которой реализует соответствующий процесс решения ССЗ.

Компонент доступа к системе (веб-компонент) представляет собой веб-приложение, посредством которого организуется управление целевой системой и предоставляется пользовательский интерфейс. Агент построения виртуальных моделей решает задачу построения и модификации графовой модели предметной области. Диспетчер анализа – модуль, выполняющий контроль полноты ПрО. Агент разрешения коллизий – программный компонент системы, который отслеживает и корректирует изменения, происходящие при модификации модели ПрО, поддерживает ее целостность и сохранность. Информационный агент – модуль добычи экспертных знаний и организации диалога между распределенными исполнителями в рамках одного проекта (задачи). Менеджер принятия решений – компонент, обеспечивающий синтез окончательного решения, то есть формирование предметной области. Из схемы видно, что все компоненты используют общий модуль доступа к данным.

В рамках данной архитектуры в настоящее время решена задача программной реализации компонента построения графовой модели динамической предметной области.

Реализация компонента построения графовой модели ССЗ

Агент построения виртуальных моделей, выделенный в архитектуре разрабатываемой системы, представляет собой программный компонент, предназначенный для обеспечения возможности построения, хранения, модификации и визуализации графовой модели предметной области ССЗ. Архитектура программного компонента представлена на рис. 2.

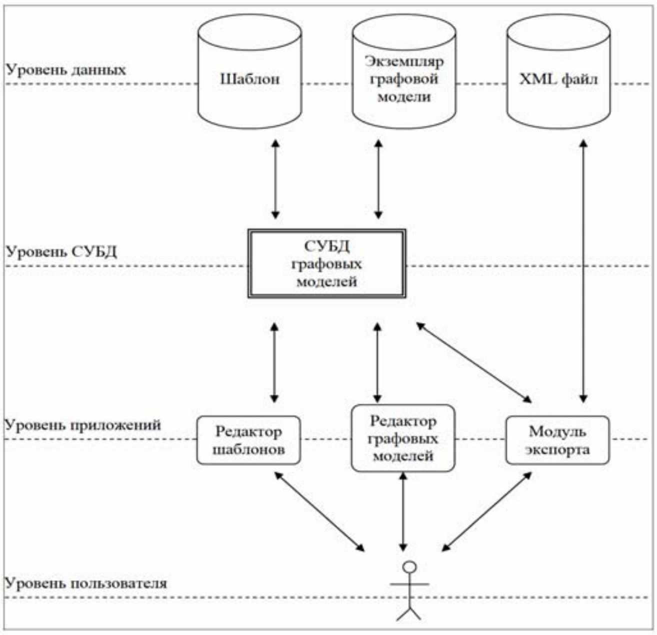


Рисунок 2 – Архитектура компонента построения графовых моделей

Разработанная и программно реализованная версия компонента позволяет решать следующие задачи:

– интерактивное создание графовой модели предметной области на основе первоначальной декомпозиции ССЗ, сохранение модели для последующего использования и модификации;

– визуализация модели;

– атрибуция вершин и дуг графа в соответствии с моделью (2);

–модификация модели за счет выполнения операций добавления, удаления вершин и дуг графа;

– разграничение доступа к фрагментам графа в соответствии с назначенными экспертами для решения подзадач;

– экспорт экземпляра графовой модели в файл формата XML.

Реализация компонента выполнена на платформе Java с использованием свободно распространяемых фреймворков и инструментов Java-окружения (Hibernate, Spring Framework, Spring Security). Выбор указанных технологий обусловлен возможностью дальнейшего расширения системы. Для визуализации графовой модели использовалась библиотека jQuery, в качестве формата для хранения модели – текстовый формат обмена данными JSON (JavaScript Object Notation). В рамках данного компонента средствами скриптового языка JavaScript реализован автономный модуль для построения и визуализации графовых моделей. Модуль позволяет визуализировать любую графовую модель, в том числе модель предметной области ССЗ, которая задается множеством узлов, определяющих подзадачи и множеством дуг, определяющих их иерархические отношения (рис. 3).

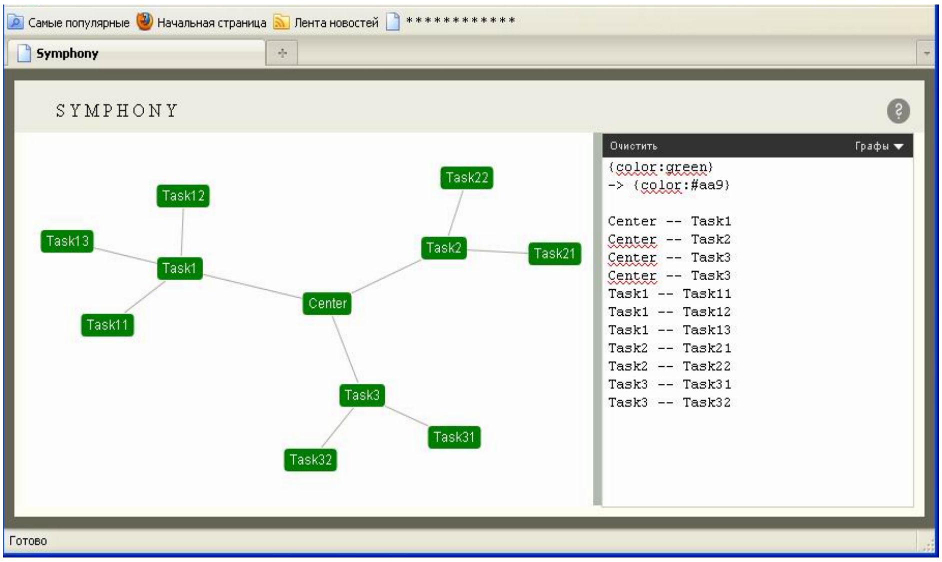


Рисунок 3 – Модуль построения и визуализации графовых моделей

Заключение

На основе анализа общепринятых стандартов описания графов и теоретико-графовых моделей, базирующихся на технологии XML, рассмотрена возможность создания языка представления динамической предметной области за счет расширения языка разметки направленных графов DGML. Представленное решение реализовано за счет включения дополнительных структурированных элементов языка и их атрибутов, отвечающих специфике предметной области ССЗ. Предложен формат представления графовой модели, позволяющий описать модель предметной области задачи (1) в виде формальной спецификации, допускающей машинное представление графа, его автоматизированную обработку и облегченный механизм синтаксического анализа.

Разработана архитектура программной системы для решения ССЗ распределенными группами исполнителей, выделены соответствующие компоненты программной системы, представлен пример реализации агента построения графовых моделей динамических предметных областей. Дальнейшее развитие проекта будет осуществляться в направлении объединения разработанных компонент в единую программную систему, а также в направлении создания средства интерактивного редактирования графой модели предметной области.

Литература

1. Вальвачев А.Н. Технология выполнения IT-проектов коллективами распределенных исполнителей / А.Н. Вальвачев, Х. Виссия, В.В. Краснопрошин // Искусственный интеллект. – 2008. – № 3. – С. 63-69.
2. Карканица А.В. Онтологический подход к построению моделей динамических предметных областей / А.В. Карканица // Вестник Гродненского государственного университета имени Янки Купалы. Серия 2. – 2010. – № 1(92). – С. 92-97.
3. Краснопрошин В.В. Алгоритмы модификации деревьев для построения динамических предметных областей / А.В. Карканица // Искусственный интеллект. – 2010. – № 4. – С. 63-69.
4. Елиферов В. Процессный подход к управлению / В. Елиферов, В. Репин. – М. : Стандарты и качество, 2005. – 408 с.

Literatura

1. Val'vachev A.N. Iskusstvennyj intellekt. № 3. 2008. S. 63-69.
2. Karkanica A.V. Vestnik Grodnenskogo gosudarstvennogo universiteta imeni Janki Kupaly. Serija 2. № 1 (92). 2010. S. 92-97.
3. Krasnoproshin V.V. Iskusstvennyj intellekt. № 4. 2010. S. 63-69.
4. Eliferov, V. Processnyj podhod k upravleniju. M.: Standarty i kachestvo. 2005. 408 s.

В.В. Краснопрошин, Г.В. Карканіца

Технологія побудови динамічних предметних областей на основі графових моделей

У статті розглядається проблема моделювання динамічної предметної області для класу задач, характер і сцена рішення яких мають розподілену структуру. Пропонується спосіб представлення графової моделі предметної області на основі розширення мови розмітки спрямованих графів. Розглядається питання проектування архітектури програмної системи для рішення складно структурованих задач. У рамках цієї архітектури пропонується реалізація програмного компонента побудови графової моделі динамічної предметної області.

V.V. Krasnoproshin, A.V. Karkanitsa

Technology of Construction of Dynamic Subject Domains Based on Graph Models

The paper is devoted to the problem of modeling of dynamic subject domains for distributed tasks. It is proposed a method of representation of the graph model of subject domain based on the eXtensible Markup Language for directed graphs. The paper discussed the problem of designing the software architecture for solving tasks with complicated structure. It is proposed implementation of the software component for construction of graph model of a dynamic subject domain.

Статья поступила в редакцию 22.06.2011.